VFAST Transactions on Software Engineering
http://vfast.org/journals/index.php/VTSE@ 2018, ISSN(e): 2309-6519; ISSN(p): 2411-6327
Volume 13, Number 1, January-April , 2018 pp. 15-21

A SURWEY OF AUTOMATED TESTING TOOLS

MUHAMMAD ADEEL KHALID [1], MUHAMMAD ALI NAEEM[2]

[1]Department of Software Engineering, University of Management and Technology, Lahore, Pakistan
[2]Department of Software Engineering, University of Management and Technology, Lahore, Pakistan
Email: {adeelkhalid992[1], malinaeem7[2]}@gmail.com

ABSTRACT. *Software applications are widely used in almost every field now-a-days. A full functional application is developed after passing through different phases of Software Development Life Cycle (SDLC), till the end user starts using it. Testing the application is one of the major tasks of Software Development Life Cycle known as SDLC. This activity is done for the effective performance, tracking out causes of inefficiencies and verifying whether a module or application fulfills the requirements. The purpose is to avoid defects, abnormal behavior, minimize risks of failure and ensure that the system is defect free. Testing can be done in both manually and automatically. Manual ways are not trust worthy because humans make mistakes and machines don't if it's programmed correctly. In this paper we have performed critical analysis on the automated testing tools available for .NET (which is a software development platform by Microsoft) determines their effects on effort, quality, productivity and cost of the product [9].*
Keywords*:* Automated testing; .NET, N-Unit, X-Unit; MS-test automated testing.

1. Introduction. Software Engineering plays important role in the development of software products. It deals with designing, development, maintenance and testing of software systems. It defines best practices; provide guidelines about what needs to be done for the software that needs to be engineered to successfully address the requirements of the customer by implementing the practices and principles effectively.

Testing activity is the most important and compulsory part of SDLC. It helps in identifying whether a system/component/module meets the expectations or not. It consumes most of the life-cycle resources i.e. Effort, Schedule, Cost etc. Testing activity can be performed in two ways; manual which is done by human and automated which is done by some testing tools. Some software organizations prefer manual testing, some prefer automated testing and some prefer both (manual as well as automated) techniques. There are reasons why software organizations prefer manual/automated testing tools/techniques [18]. In manual testing; a method for running tests manually by creating manual scenarios for the system that needs to be build/tested against its recommended functionality. Human can think more scenarios while performing testing. A continuous testing, testing doesn't stop on errors.

Automated testing is useful for large systems. It reduces manpower invested in the testing phase. This results in reducing the cost consumption on this phase and increase productivity. Most automated testing tools break the execution whenever an error is detected. The motive behind these tools is to fix the first problem first and then move on. It cannot get tired like humans do. Software organizations especially new/immature ones, face issues in selection of appropriate platform for the software development. There are number of factors that swirls around this selection i.e. scope of the project, programming language i.e. C-Sharp development team expertise, development tools like IDE's, DBMS like MS-SQL Server, Oracle etc… testing tools like X-Unit, N-Unit, MS-Test etc… and versioning control systems like Visual SVN, GIT and so on [14][15][16].

There were many tools available related to the testing, some were related to the unit testing, some were related to the exploratory testing, some were related to regression testing, some were related to the web

development like cross browser compatibility checking, some were related to the android development so on and so forth. There are many discriminating factor behind the existence of automated testing tools that can really influence the project performance, cost, effort, schedule, productivity are time and cost, these are the most important factors. If you can't complete a project within approved budget and within allocated time then it's going to fail anyway. The usefulness of testing tool can be evaluated by checking its ability to capture errors/defects inserted intentionally into the unit of code.

In this paper, we have put our effort in analyzing the best testing tools for unit testing available in C-Sharp. We gathered the information related to the available testing tools from various sources, firstly we collect data from development related QA websites. We've focused on questions where people ask something related to the automated testing tools like how automated tools work and what are the issues they are currently facing while using those tools. Secondly we've analyzed the currently job requirements for the post of Quality Assurance department, we've observed what sort of experience the companies ask, and what type of automated testing tools they are interested in.

2. Literature reviw: In order to start automated software testing, we need to use automated testing tools which record use cases and start testing. For running automated testing we need to input a number of required test cases in an automated testing tool. After giving input of test cases the tool is responsible for performing the software tests. Testing activity consumes almost half of the resources of the SDLC [1]. Automated testing helps to reduce the cost by automating the test activity.

The use of automated testing supported by the factors like reusability [2], less human interaction and stable technology support. Tester used tools for software testing projects. So, Automation is the main limited to test execution and defect reports. Test cases are composing arbitrarily and instruction are needed for tester [3]. If the testing through tools is not properly conducted or planned, it would not reduce the cost related to the testing phase and would consume the same amount of cost in terms of resources like the manual testing do [4]. On the observation of `T. Repo` [2], the main benefit of automation testing includes quality, test coverage and much less time done in testing. Human involvement is very important in selection of test cases; observations are made to get a better test coverage.

The focus of automated testing is to eliminate or reduce the issues/problems faced by the tester by the automated testing by providing the test cases [5]. X-Unit is the nonexclusive name for the group of unit test structures that are currently available in relatively every programming language i.e. Java, c#, MS-Test, N-Unit is also one of them [6][7].

Reusability and prioritization of test cases can increase productivity because same tests could be performed with a number of changed scenarios. There exists a number of Software Testing Automation Frameworks that helps in reusing services that can automate testing process activities which is a multi-language, multiplatform technique for reusing. Testing tools increase reliability by reducing the risk of system failures without involving human because a human can make errors but machine cannot make error until it is programmed to do so [8].

3. Methodology. There are many platforms available today for software development. We've considered C-SHARP which is well known as C# or C-HASH of .NET. The selection of this platform depends on various factors like the platform support desktop development, web development, mobile development, API development; in short it provides all the flavors in one plate. The usages of this platform are very vast. For the sake of our research, we've performed the following steps are performed.

- **Selection of automated testing tools:** Before directly digging into the tools, one must need to understand that it is necessary to write down a list of requirements that needs to be tested and then get the tool that can work with all/partially those requirements. Using a testing tool without understanding the requirements is just the waste of time and resources. In short, we must understand the nature of the application under testing. As we are dealing with unit testing, we've considered the tools related to the unit testing. There are numerous testing tools available for performing unit testing in .NET. Some of them are open source, some are premium, some tools support more than one platform and some are platform dependent. A detailed study has been conducted on various testing tools available for .NET. We've examined their pros and cons carefully, analyzed their performance, and judged impact on

overall cost of the project in terms of effort, risks, quality and so on. We've outlined criteria under which we've selected the testing tools.

- **Criteria for the tools selection:**

Source	***Analysis***
Stack overflow Analysis	Question answers related to the automated testing tools in C-Sharp
Testing Jobs	Indeed and Rozee have been analyzed for the Testing jobs to assess the testing tools
Web Search	Analyzed the tools ranking in Google search when specific tool name is given as a query

TABLE-1: SELECTION OF TOOLS.

- **Testing tools for C-Sharp:** As .NET has become open source, all its developed, supported testing frameworks now turned into cross platform which means all the testing frameworks and developed applications of Microsoft can now run, deployed on MAC, Linux, Windows and others at the same time. The testing tools can also work on these platforms with the IDE (i.e. Visual Studio Code or Visual Studio Community and so on) [17]. C Sharp is an extensively used programming language in the company. It is a programming language that relies on Microsoft's programming language. If you are developing an application with C Sharp it is possible to use Visual Studio and explore some extensions to improve development.
    - **N-Unit:** Testing framework developed by Charlie Poole and others. The maker of N-Unit has joined .net Foundation so it could be improved with better assistance, expertise and tools. It supports all .NET languages i.e. C#, VB.Net and F#. It is reliable, fast, and easy to integrate with off-the-shelf tools, open source and many developers contributed in its magic run. A simple class needs to be developed for performing unit test within the under development project for unit testing [14].

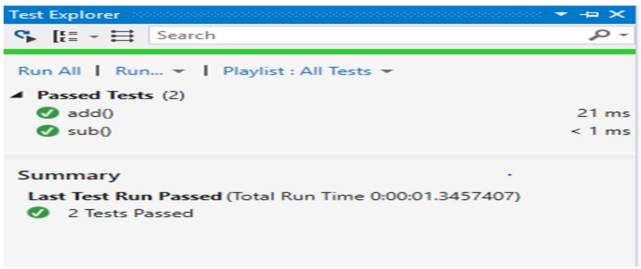

Figure-1: Execution of N-Unit.

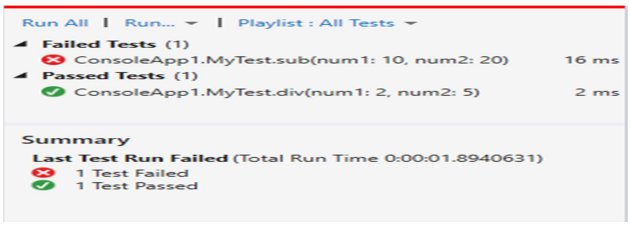

Figure-2: Execution of X-Unit.

- **X-Unit:** Testing framework developed by the inventors of N-Unit. Its free, open source tool that works with .net technologies. Its sole purpose is to perform unit testing. It's about dividing tests into theories and facts to differentiate between "true" and "always true" [15].
- **MS-Test:** It is also developed and supported by the Microsoft, like N-Unit, supports all .NET languages too. It is tightly integrated with Visual Studio (an Integrated Development Environment) by Microsoft. It automatically generates unit tests and bug linking in TFS (Team Foundation Server). A test project needs to be developed and all the Dynamic Link Libraries (DLL) needs, which need to be tested, are imported into that project to perform unit test within Visual Studio, from a separate project [16].

Figure-3: Execution of X-Unit.

Tool	*Experience Required*	*Installation*
MS-Test	Basic	Shipped with IDE
X-Unit	Average	Separate installation
N-Unit	Average	Separate installation

TABLE-2: SKILLS AND INSTALLATION.

The table shows the difference in terms of skill set required to implement the test in the above mentioned testing tools and the installations for them. X-Unit and MS-Test require Average skills because X-Unit implements Fact and Theories which needs to understand what they are about, MS-Test require basic skillset, it can be used directly from the main project by just writing the test cases or you can do this in a separate MS Test project where you can track what you have tested so far and what is left to be tested.

N-Unit require average knowledge, as it's the pioneer testing tool available on .NET platform, most of the people are familiar with it and the implementation doesn't require a lot of expertise, the pain is hidden while making its setup. Another fact of difference is the installation. Only MS-Test is the one testing tool that is shipped with the Integrated Development Environment i.e. Visual Studio. X-Unit and N-Unit are installed separately via Nuget Package Manager (a package manager by Microsoft shipped with Visual Studio). One time installation of either of them is enough to perform long-term testing with them [12] [13].

4. Results.

- **Comparison.** We've created three different test cases for the purpose of experiments for conducting tests on both the platforms i.e. C-Sharp run on all their respective testing tools. First test case was a simple calculator which does add, subtract, multiply and divide respectively. Second test case include registration form, where user give user name and password, email, phone number, age, sex and the system validates whether all the input data is correct or not. Third test case performs some calculations of a booking system where the cost of user booking is calculated based on the number of travelers, days, discount and tax.

  For test experiments we've used the machine i5 6th generation, 8 GB RAM. There could be a tradeoff among the system specification and the testing tool performance; the recorded performance could be change when the tests are performed on machine with high specs or on the machine with lower specification. Although all three tools are good enough for performing unit testing, there are no standard defined or guidelines available to perform comparisons between them. So, we noticed the minor difference when started developing the test cases for all of these tools, we first noted that LOC is

varying for MS-Test, while for X-Unit and N-Unit, the number was same.

Another thing we noticed that MS-Test is executed in a separate project while X-Unit and N-Unit can execute directly from the main project there is no need to execute them from another project. We need to include the libraries of our code in MS Project in order to test using it while in other tools, it is not required thing.

- **Evaluation:** The testing tools differ in terms of the time it take to develop a test case, LOC for each test to perform and the time it take to execute the test which is the most important factor that can affect the development cost, effort invested on the manual testing. We've analyzed all the tools with these 3 parameters and noted the difference. While running the tests for the first time it showed slight difference in terms of LOC, Development and Execution Time.

  When we ran the same tests 10 times, the result showed a great difference in the execution time. N-Unit took very little time in execution as compared to X-Unit and MS-Test. Based on the results N-Unit caught our attention and we rank it first based on the fact that it consumes very little time in executing the test cases while X-Unit on the second and MS-Test on the third place. Our results show that if we start using some tool for the Microsoft framework for unit testing we should consider N-Unit as our first priority X-Unit on the second the MS-Test on last among these three. The following table shows the difference we've noted.

Tool Name	**Test Cases**			
	TC-1	***TC-2***	***TC-3***	***TC-4***
N-Unit	5	5	10	12
X-Unit	5	5	10	12
MS Test	6	8	12	17

TABLE-1: TEST CASE DEVELOPMENT TIME.

Tool Name	**Test Cases**			
	TC-1	***TC-1***	***TC-1***	***TC-1***
N-Unit	60	110	230	270
X-Unit	60	110	230	270
MS Test	69	120	250	320

TABLE-2: LINES OF CODE FOR EACH TOOL.

Tool Name	**Test Cases**			
	TC-1	***TC-2***	***TC-3***	***TC-4***
N-Unit	36	69	61	65
X-Unit	66	62	82	96
MS Test	97	102	121	137

TABLE-3: TEST CASE EXECUTION TIME IN MILLISECONDS.

Iterations	**Tools**		
	N-Unit	***X-Unit***	***MS-Test***
IT-1	36	66	97
IT-2	69	62	102
IT-3	61	82	121
IT-4	65	96	137
IT-5	68	98	122
IT-6	64	90	105
IT-7	66	102	111
IT-8	60	94	109

TABLE-4: TEST CASES EXECUTION IN MILISECONDS.

5. Critical analysis of results. A software firm invests more money in manual testing which can be reduced by using the automated testing tools. The feedback from the people using automated testing tools can improve the testing tools so that the need of manual testing can be limited to some high level extent and most of the testing will be done using automated testing tools to save the time, cost, and resources. Vendors of automated testing tools claim that their tools cover most of the requirements of the manual testing and are better than others existing automated tools but this is not true. Vendors make false statements in order to sale, promote their product although there are fewer success stories behind the real usage and benefits of their testing products. We analyzed these false statements when we critically compared those tools. While starting the testing all were showing the same level of performance but when repeated and changed the test cases those testing tools started making difference in their level of performance, time, LOC.

Software industry should need to understand the benefits of using automated testing tools. They already aware with the benefits but they don't want to take risk and change their traditional way of testing (converting manual testing into automated testing) because they think that this transition will charge them more in terms of training their developers, testing teams according to these tools. Its one-time cost of training.

6. Conclusion and future directions. This work is related to the unit testing. The tools which support unit testing are considered and avoided the tools that support other forms of testing. We've performed experiments on the top level unit testing tools available for C-Sharp and tried to capture the difference in terms of performance measure taken to perform the experiments. We cannot say a tool is better after first run, the main difference comes when we repeatedly play some test and observe the change in their execution, this way the results could draw our attention to some major difference. If we do it right, it can be very much beneficial for the overall cost, quality and performance of the project.

We conclude that N-Unit is the best tool among all three. The next phase of this paper would deal with the selection of open source testing tools for java and python, object oriented test cases for object identification using reflection, entity framework model utilization, ranking those tools based on their associated attributes like usefulness, recording of test cases, efficiency, reuseability of test cases, sync test cases and their results with some online repository to make them use in future projects and many other things shows huge gap in this field which needs to be filled slowly, gradully and with accuracy of authentic results which can be best fit and thought out for further investigation in this field[10][11].

REFERENCES

[1] Ramler, R., & Wolfmaier, K. (2006, May). Economic perspectives in test automation: balancing automated and manual testing with opportunity cost. In *Proceedings of the 2006 international workshop on Automation of software test*(pp. 85-91). ACM.

[2] Karhu, K., Repo, T., Taipale, O., & Smolander, K. (2009, April). Empirical observations on software testing automation. In *Software Testing Verification and Validation, 2009. ICST'09. International Conference on* (pp. 201-209). IEEE.

[3] Li, K., & Wu, M. (2006). *Effective software test automation: developing an automated software testing tool*. John Wiley & Sons.

[4] Amannejad, Y., Garousi, V., Irving, R., & Sahaf, Z. (2014, March). A search-based approach for cost-effective software test automation decision support and an industrial case study. In *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on*(pp. 302-311). IEEE.

[5] Amannejad, Y., Garousi, V., Irving, R., & Sahaf, Z. (2014, March). A search-based approach for cost-effective software test automation decision support and an industrial case study. In *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on*(pp. 302-311). IEEE..

[6] Meszaros, G. G. (2010, October). XUnit test patterns and smells: improving the ROI of test code. In *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*(pp. 299-300). ACM.

[7] Crookshanks, E. (2015). Unit Testing and Test-Driven Development. In *Practical Enterprise Software Development Techniques* (pp. 91-107). Apress, Berkeley, CA.

[8] Ramamoorthy, C. V., & Ho, S. B. F. (1975, April). Testing large software with automated software evaluation systems. In *ACM SIGPLAN Notices* (Vol. 10, No. 6, pp. 382-394). ACM.

[9] .NET Framework, (2018, Mar). [online] Available: https://www.microsoft.com/net/learn/what-is-dotnet.

[10] Marcozzi, M., Vanhoof, W., & Hainaut, J. L. (2015). Relational symbolic execution of SQL code for unit testing of database programs. *Science of Computer Programming*, *105*, 44-72.

[11] Xie, T., Tillmann, N., & Lakshman, P. (2016, May). Advances in unit testing: theory and practice. In *Proceedings of the 38th International Conference on Software Engineering Companion*(pp. 904-905). ACM.

[12] Nuget Package Manager, (2018, Mar). [online] Available: https://www.nuget.org/.

[13] Visual Studio, (2018, Mar). [online] Available: https://www.visualstudio.com/.

[14] NUnit, (2018, Mar). [online] Available: http://nunit.org/.

[15] XUnit, (2018, Mar). [online] Available: https://xunit.github.io/docs/getting-started-desktop.html

[16] MS Test, (2018, Mar). [online] Available: https://docs.microsoft.com/en-us/dotnet/core/testing /unit-testing-with-mstest.

[17] .NET CORE, (2018, Mar). [online] Available: https://blogs.msdn.microsoft.com/dotnet/2018/02/27/ announcing-net-core-2-1-preview-1/.

[18] Alshamrani, A., & Bahattab, A. (2015). A comparison between three SDLC models waterfall model, spiral model, and Incremental/Iterative model. *International Journal of Computer Science Issues (IJCSI)*, *12*(1), 106.

[19] Khan, Y. D., Ahmad, F., & Anwar, M. W. (2012). A neuro-cognitive approach for iris recognition using back propagation. World Applied Sciences Journal, 16(5), 678-685.

[20] Khan, Y. D., Khan, S. A., Ahmad, F., & Islam, S. (2014). Iris recognition using image moments and k-means algorithm. The Scientific World Journal.